Transaction Systems
Exercise Session 12

Andrey Gubichev

February 3, 2014
Homework, Task 1

\[ s_1 = w_1(x)w_1(y)r_2(u)w_2(x)r_2(y)w_2(y)a_2w_1(z)c_1 \text{ (PRED)} \]

\[ s_2 = w_1(x)w_1(y)r_2(u)w_2(x)r_2(y)w_2(y)w_1(z)a_1c_2 \text{ (not PRED)} \]
Homework, Task 2

Which of the properties RC, ST, RG, PRED, and LRC are satisfied by the following schedules:

- $s_1 = r_1(a)r_2(a)w_1(a)c_1c_2$ (RC, ST, not RG, PRED, LRC)
- $s_2 = r_1(a)w_1(a)r_2(b)w_2(b)w_2(a)c_2c_1$ (RC, not ST, not RG, not PRED, not LRC)
- $s_3 = r_1(a)incr_2(a)incr_2(b)incr_3(b)c_3a_2c_1$ (not RC, not ST, not RG, not PRED, not LRC)
Homework, Part 2

- Determine the necessary log actions (during normal work) and recovery actions (during restart). Redo-history

<table>
<thead>
<tr>
<th>Action</th>
<th>Change of cached DB [PageNo:SeqNo]</th>
<th>Change of stable DB [PageNo:SeqNo]</th>
<th>Log added to buffer [LogSeqNo:SeqNo]</th>
<th>Log added to Log [LogSeqNo’s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: begin($t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: write($p$, $t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: write($q$, $t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: commit($t_1$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: flush($p$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6: begin($t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7: write($p$, $t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8: write($r$, $t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9: checkpoint</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10: commit($t_2$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11: begin($t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12: flush($p$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13: write($p$, $t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14: write($q$, $t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15: flush($q$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16: write($r$, $t_3$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... system crash</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Homework, Part 2, Cont.

- First consider the case where heavyweight checkpoints are used and flush actions are not logged; then consider lightweight checkpoints; finally, discuss the additional effect of keeping flush actions
(Basic) Logging rules

- When committing, flush the log to stable media (Redo)
- Page flush first forces the log buffer to the stable log (Undo)
- GC: log of committed transactions is kept until all their modified pages are written to stable storage
Redo-Winners vs Redo-History

- R-W: redo only the transactions that were completed (winners), undo losers
- R-H: redo everything, undo losers
- R-H: keep the compensation actions logged
Physical vs Physiological writes

- Physical actions: write the full page (idempotent)
- Physiological actions: perform logical operation on physical pages (non-idempotent)
- Difference is in how we do logging
- Why is it important? Physical operations can be redone-undone many times
Log truncation

- Smaller log means faster recovery
- SystemRedoLSN: oldest write operation among all dirty pages
- OldestUndoLSN: seq number of the write operation among all active transactions
- min among them is the new starting pointer
Checkpoints

- Redo needs to repeat the entire log, unless we keep track of relevant transactions
- Heavy checkpoint: flush all pages from cache, keep the list of active transactions
- Light checkpoints: keep the list of dirty pages and the corresponding LSN
Flushing log entries

- page not included to checkpoint, but updated and flushed later on
- can flush the log for these entities. Speed up the redo part
Undo completion

- to avoid undoing the compensation actions, keep NextUndoSeqNo for all entries
- can flush the log for these entities. Speed up the redo part
Exam To-Do list

1. understand main concepts thoroughly (ideally with details, relationships between them etc)
2. learn algorithms, pay attention to their applicability and pros/cons. Special attention to basic algos
3. ability to construct examples is quite useful
4. no need to learn the proofs from the lecture/book/exercises (esp. if you can prove them on the fly) :) 
5. we may ask you to prove some (very simple) statements (see 1. and 2.)
Exam structure

- 6-8 tasks
- at most one question to prove smth
- some tasks are to apply algorithms
- some tasks are about classes of histories
Examples

Question 1: for the two given schedules list all the classes they belong to from the following list of classes: Serial, CSR, VSR, FSR, RC, ACA, ST, RG, PRED, LRC. Every correct class is +1 point, every incorrect class is -1 point:

\[ s = w_1(x)w_2(x)c_2c_1 \]
Question 2: for the following schedule give the output schedules produced by two protocols: (a) SS2PL, (b) BTO:

\[ s = r_1(x)w_2(x)r_3(y)w_2(y)w_3(z)c_3r_1(z)c_1r_2(z)c_2 \]

What is the protocol (not necessarily mentioned above) that accepts this schedule as is (without delaying or aborting anything)?
Examples

- Give an example of a schedule that
  - is conflict serializable, but not log recoverable
  - is strict but can not be generated by Strict 2PL
Question 5: Describe the transactional workload for which the following protocol will provide the best throughput:

- (among all page model protocols) FOCC
- (among all object model protocols) a combination of 2PL at the object level and FOCC at the page level

Give a short explanation
Info

- Exam on February 10, at 2.30 PM
- Location: MW 0350, Egbert-von-Hoyer-Hörsaal